Reasoning with Inclusion Axioms in Description Logics: Algorithms and Complexity

Diego Calvanese[1]

Abstract. The computational complexity of reasoning on pure concept expressions has been characterized completely for all relevant description logics. On the contrary, reasoning in the presence of schema axioms is not so well understood and far from being settled completely. An important class of schemata is that of primitive schemata (in which the schema axioms express only necessary conditions) possibly containing cycles. In this paper we provide, for a relevant class of description logics, a complete characterization of computational complexity of reasoning in these types of schemata, both in the presence and in the absence of cycles. The results are obtained by devising reasoning procedures, establishing direct reductions to show lower bounds, and introducing a general technique by which the constructor for existential quantification can be removed without influencing the result of reasoning.

1 INTRODUCTION

Description logics were originally introduced as a formalization of Frame based systems and Semantic Networks. They allow one to express structured knowledge by means of concepts and roles (which represent sets of objects and binary relations respectively) and provide mechanisms to reason on such structured descriptions. The typical reasoning tasks considered are: Determining if a concept is consistent (i.e. does not necessarily denote the empty set) and determining subsumption between concepts (i.e. if one concept necessarily denotes a subset of another one).

Initially, research concentrated on the restricted problem of reasoning on concept expressions, without taking into account a schema containing concept definitions and/or explicit inclusion assertions. The study of this problem was motivated by the following observation: Under the assumption that the axioms in the schema contain no cyclic references between concepts, the problem of concept subsumption with respect to such a schema can be reduced to subsumption between pure expressions. This is achieved simply by unfolding the axioms in the schema. Although this can in the worst case lead to an exponential increase in the size of the concepts [16], this seems not to happen in practical cases. The computational complexity of reasoning on concept expressions has been exactly characterized for all relevant description logics [14, 19, 15], and this problem can now be regarded as solved.

However, although the assumption of acyclicity is common to many terminological systems, it is unrealistic in practice, and cycles are definitely necessary for a correct modeling in many application domains. This is further confirmed by the fact that their use is allowed in most database models. Cycles not only require a special treatment by reasoning procedures and greatly increase the complexity of reasoning [1, 2, 17], but raise also the problem of which semantics to adopt to interpret them [17]. So called *descriptive semantics* accepts all interpretations that satisfy the axioms. For this reason, under descriptive semantics a cyclic schema does not provide definitions, and alternative semantics, which are based on fixpoint definitions have been proposed to overcome this problem [17, 18, 12]. Following [4] we argue, however that a schema represents a set of constraints on the domain to be modeled, and therefore we should adopt descriptive semantics which interprets axioms precisely as constraints.

These considerations motivate also the interest in reasoning on *primitive schemata*, i.e. schemata containing only primitive concept specifications (which state just necessary conditions) and no definitions (which state both necessary and sufficient conditions). In fact, the constraints that can be expressed in most semantic and object-oriented data models correspond naturally to primitive concept specifications, and as shown in [10, 3], reasoning in these formalisms can correctly be captured by reasoning on primitive schemata.

Deduction on schemata is not so well understood and far from being settled in all relevant cases. Cyclic definitions are investigated in [1, 17], where for a simple logic containing only conjunction and universal quantification, subsumption in possibly cyclic schemata is shown to be already **PSPACE**-complete for fixpoints semantics and in **PSPACE** for descriptive semantics (In fact, in [17] more expressive logics are considered). For much more powerful logics **EXPTIME**-completeness of schema level reasoning is established in [11, 8]. Primitive schemata, on the other hand, have been investigated only recently, motivated by the tight correspondence with database models. The impact of the assumption that the schema contains no definitions is not well understood. In particular, the known reasoning procedures cannot take advantage of the fact that the schema is primitive and treat it as an arbitrary schema possibly containing definitions. Also, the known lower bounds are the ones that follow trivially from the lower bounds established for reasoning on concept expressions. Exceptions are the results established in [6, 10, 4], which deal explicitly with primitive schemata. We discuss the results in [6, 4] relevant for this paper in Section 3. [10] shows decidability in deterministic exponential time of reasoning on primitive schemata in an expressive logic that allows to represent database models and frame based systems.

In this paper we present a systematic study of the complexity of deduction on primitive schemata expressed in the basic description logics of the $\mathcal{AL}$-family [15], namely $\mathcal{ALC}$ and its sub-languages. To this end we provide three distinct technical contributions: (1) We develop an algorithm for reasoning in polynomial space in acyclic schemata in the presence of disjunction. (2) We show various lower bounds by direct reductions. (3) We introduce a general technique by which the constructor for existential quantification can be eliminated from schemata without influencing the result of deductions. The com-

[1] Dipartimento di Informatica e Sistemistica, Univ. di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy. calvanese@dis.uniroma1.it

ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

DL\Schema	empty	acyclic	general
$\mathcal{FL}^-$ (subsum.)	**PTIME**	**PTIME**	**PTIME**
$\mathcal{AL}$	**PTIME**	**coNP**	**PSPACE** $\triangle$
$\mathcal{ALE}$	**coNP**	**coNP** $\triangledown$	**PSPACE** $\bigcirc$
$\mathcal{ALU}$	**NP**	**PSPACE** $\bigcirc$	**EXPTIME** $\triangle$
$\mathcal{ALC}$	**PSPACE**	**PSPACE** $\triangledown$	**EXPTIME**

Table 1. Complexity of concept consistency in primitive schemata

bination of these three contributions allows us to characterize completely the complexity of reasoning in all considered logics, with respect to both acyclic and general schemata. Section 2 provides the notational and definitional background, while the technical results are presented in Section 3.

2 BACKGROUND

Description logics allow one to express structured knowledge by means of *concept* and *role expressions*. Concept expressions (also called simply *concepts*) can be considered as unary predicates which are interpreted as sets of objects, whereas *roles* are binary predicates interpreted as binary relations over the domain of objects. Complex concepts and roles (ranged over by C and R respectively) can be built starting from a set of *concept names* (ranged over by A, B) and *role names* (ranged over by P) by applying the constructors that characterize the logic. A systematic presentation of the class of description logics we deal with in this paper, namely $\mathcal{AL}$-logics, can be found for example in [15], and we refer to this paper for the denotation of the logics and formal definitions of syntax and semantics.

Given a description logic $\mathcal{L}$, an $\mathcal{L}$-schema introduces concept and role names and states conditions that objects have to satisfy in order to be instances of certain concepts. Formally, an $\mathcal{L}$-schema $\mathcal{S}$ is a triple $(\mathcal{A}, \mathcal{P}, \mathcal{T})$, where $\mathcal{A}$ is a set of concepts names, $\mathcal{P}$ a set of role names, and $\mathcal{T}$ a set of *axioms* that have the form

$$\begin{array}{lcll} A & \dot{\preceq} & C & \text{(primitive concept specification)} \\ A & \doteq & C & \text{(concept definition)}, \end{array}$$

where $A \in \mathcal{A}$, C is a concept containing only names in $\mathcal{A} \cup \mathcal{P}$, and for each concept there is at most one definition. A schema containing only primitive concept specifications is called *primitive*. A cycle in a schema is defined as follows. We say that A *directly uses* B if for some axiom $A \dot{\preceq} C$ or $A \doteq C$ in $\mathcal{T}$, B appears in C. Let *uses* denote the transitive closure of "directly uses". Then $\mathcal{S}$ contains a *cycle* if there is a concept name that uses itself. A schema possibly containing a cycle is called *general*.

An axiom $A \dot{\preceq} C$ states necessary conditions for an object to be an instance of A, while $A \doteq C$ states both necessary and sufficient conditions. Formally, an interpretation $\mathcal{I}$ *satisfies* $A \dot{\preceq} C$ (resp. $A \doteq C$) if $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ (resp. $A^{\mathcal{I}} = C^{\mathcal{I}}$), and an interpretation of $\mathcal{S}$ that satisfies all axioms in $\mathcal{T}$ is called a *model* of $\mathcal{S}$. A concept C is said to be *consistent in* $\mathcal{S}$ if $\mathcal{S}$ admits a model $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$, and a concept C is *subsumed by* a concept D *in* $\mathcal{S}$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{S}$.

Notice that we have adopted *descriptive semantics*, which seems to be the most appropriate one for the interpretation of schemata [4, 17]. In the following we assume to deal only with primitive schemata.

3 COMPLEXITY RESULTS

In this section we analyze the complexity of reasoning on *primitive* schemata, both acyclic and general, expressed in $\mathcal{ALC}$ and its relevant sub-logics. The results we establish allow us to fill in all entries in Table 1 that were unsettled, obtaining tight complexity bounds in all cases. In particular, the completeness (resp. upper bound, lower bound) results for entries marked with "$\bigcirc$" (resp. "$\triangledown$", "$\triangle$") were open and are established here. We have also included for comparison a column showing the complexity of pure consistency, i.e. of consistency of concept expressions with respect to an empty schema.

For $\mathcal{FL}^-$-schemata concept consistency is trivial, since $\mathcal{FL}^-$ contains no combination of constructs that can give rise to a contradiction. Therefore any concept in an $\mathcal{FL}^-$-schema is consistent. An immediate consequence of the polynomial algorithm for query subsumption given in [6] is that concept subsumption in $\mathcal{FL}^-$ can be decided in polynomial time even with general schemata.

In the following, for lack of space, we concentrate on concept consistency only. However, it is possible to show that the corresponding complexity bounds hold also for the case of subsumption. We assume that schemata are *normalized*, i.e. all concepts that appear on the right hand side of the axioms in the schema are of the form $A \mid \neg A \mid A \sqcup B \mid \forall P.A \mid \exists P \mid \exists P.A$. This represents no limitation since we can bring all concepts in negation normal form, and by introducing at most a linear number of new concept names and axioms, eliminate conjunction and the nesting of constructors without influencing consistency. Moreover, we can restrict our attention to the case where the concept to check for consistency is a single concept name, since a generic concept C is consistent in a schema $\mathcal{S} := (\mathcal{A}, \mathcal{P}, \mathcal{T})$ iff A is consistent in $\mathcal{S}' := (\mathcal{A} \cup \{A\}, \mathcal{P}, \mathcal{T} \cup \{A \dot{\preceq} C\})$, where A is a new concept name.

3.1 Direct upper bounds

In [4] an algorithm is given for verifying concept consistency in schemata in a slight extension of $\mathcal{AL}$. We briefly sketch this method for the case of $\mathcal{AL}$-schemata. First of all, notice that since all concepts in the schema are normalized (and there is no disjunction), we can assume that the schema is *ISA-complete*, i.e. if we view "$\dot{\preceq}$" in the axioms as a partial order between concept expressions, then the axioms are closed under transitivity. Given such a schema $\mathcal{S} := (\mathcal{A}, \mathcal{P}, \mathcal{T})$, the algorithm is based on the construction of a graph whose nodes are subsets of $\mathcal{A}$ and whose edges are labeled by roles in $\mathcal{P}$. There is an edge labeled with P from a node N to a node N' iff

1. there is a concept $A \in N$ such that $A \dot{\preceq} \exists P$, and
2. $N' = \{A' \mid A \dot{\preceq} \forall P.A' \in \mathcal{T}$ for some $A \in N\}$.

If for a node N condition (1) is verified we say that P is *active* for N. A node is said to be a *conflict node* if it contains concept names A, B such that for some $A' \in \mathcal{A}$, $A \dot{\preceq} A'$ and $B \dot{\preceq} \neg A'$ are in $\mathcal{T}$.

Such graph can be put in correspondence with an interpretation as follows: A node corresponds to an object that is an instance of all concepts in the node. An edge labeled by P between two nodes N and N' represents the fact that in a model of $\mathcal{S}$ the object corresponding to N necessarily has a P-successor which corresponds to N', i.e. is an instance of all concept names in N'. Therefore, the conjunction of concepts in a node N is consistent iff in the graph there is no path from N to a conflict node. The entire graph is clearly exponential in the size of the schema, but a path to a conflict node can be detected by a nondeterministic algorithm using only polynomial space by computing a successor node directly from the schema and the current node. Moreover, for an acyclic schema the length of a path is bounded linearly by the number of concept names. This shows that concept consistency is in **coNP** for acyclic $\mathcal{AL}$-schemata and in **PSPACE** for general $\mathcal{AL}$-schemata.

```
consistent(N: set of concept names, S_U: ALU-schema): boolean
  begin
    i ← 0;
    repeat
      i ← i + 1;
      S ← isa_complete(choice(i,S_U));
      Consistent ← consistent_choice(N,S,S_U)
    until Consistent or last_choice(i,S_U);
    return Consistent
  end;
consistent_choice(N: set of concept names,
                  S: AL-schema, S_U: ALU-schema): boolean
  begin
    ConsChoice ← no_conflict(N,S);
    if ConsChoice then
      for each P in active_roles(N,S) do
        ConsChoice ← ConsChoice and
                     consistent(successors(P,N,S), S_U)
    return ConsChoice
  end;
```

Figure 1. Concept consistency in acyclic $\mathcal{ALU}$-schemata

We now show how the reasoning technique for $\mathcal{AL}$ can be extended to deal also with disjunction, providing an algorithm that decides concept consistency in acyclic $\mathcal{ALU}$-schemata in polynomial space. The idea underlying our algorithm is that in the presence of disjunction one can traverse the graph by making nondeterministic choices at each node. The choice involves deciding, for each axiom containing a disjunction on the right side, which of the disjuncts to consider for determining the ISA-completion. This determines also if the node is a conflict node, and the successor nodes for each role. Given a node N representing a conjunction C_N of concept names, if there is a choice such that N is not a conflict node (with respect to the ISA-complete schema determined by the choice), and the successor nodes for all roles are themselves consistent, then C_N is consistent in the schema.

Figure 1 shows a procedure that implements this idea. It takes as input a set N of concept names and an acyclic-$\mathcal{ALU}$-schema $\mathcal{S}$ and answers **true** iff the conjunction of concepts in N is consistent in $\mathcal{S}$. Procedure *consistent* tries the 2^n choices in succession (where n is the number of axioms involving disjunction), generating for each $\mathcal{AL}$-schema returned by *choice* the ISA-completion, and then calling *consistent_choice* to verify if the choice is the one that allows to satisfy the conjunction of concepts in N. Observe that *isa_complete* works in polynomial time, since the schema returned by *choice* contains no disjunctions. Procedure *consistent_choice* verifies first if N is a conflict node (with respect to the schema obtained from the choice), and then generates for each role active for N the successor node, and verifies recursively if this node is consistent.

Proposition 1 *Let $\mathcal{S}$ be an acyclic $\mathcal{ALU}$-schema and $C := A_1 \sqcap \cdots \sqcap A_n$ a conjunction of concepts is $\mathcal{S}$. Then C is consistent in $\mathcal{S}$ iff consistent$(\{A_1, \ldots, A_n\}, \mathcal{S}) =$ **true**.*

Since the schema is acyclic we can apply the same argument used in [4] to show that the length of a chain of successor nodes is linear in the size of $\mathcal{S}$. Therefore, when executing procedure *consistent*, we have at most a linear number of nestings of recursive calls, and the space taken up by the activation stack is polynomial.

Theorem 2 *Concept consistency in acyclic $\mathcal{ALU}$-schemata can be decided in worst case polynomial space in the size of the schema.*

3.2 Direct reductions to show lower bounds

As already mentioned, in [4] it is shown that when moving from $\mathcal{FL}^-$ to $\mathcal{AL}$ (which allows to express inconsistent concepts) verifying concept consistency becomes intractable. We come back in Section 3.3 to the idea underlying this result and want first to sketch an alternative proof of **coNP**-hardness for concept consistency in acyclic $\mathcal{AL}$-schemata, which is based on a direct reduction from validity of propositional formulae in DNF. The idea introduced in this reduction can also be immediately exploited to give a direct proof of **PSPACE**-hardness of concept consistency in acyclic $\mathcal{ALU}$-schemata. More important, the insight gained in this way is fundamental to show **PSPACE**-hardness of reasoning on general $\mathcal{AL}$-schemata.

We first give some definitions and discuss some simple properties of $\mathcal{AL}$-schemata. A sequence $K := A_0 A_1 \cdots A_n$ of concept names is a $(P_1 \cdots P_n)$*-chain in* $\mathcal{S}$, if $\mathcal{T}$ contains the axioms $A_{j-1} \doteq \forall P_j.A_j$, for $j \in 1..n$. A $(P \cdots P)$-chain is called a P-chain. Let $K_i := A_{i0} A_{i1} \cdots A_{in}$, $i \in 1..m$, be $(P_1 \cdots P_n)$-chains in $\mathcal{S}$. The set $H := \{K_1, \ldots, K_m\}$ is *active in* $\mathcal{S}$, if for each $j \in 1..n$ there is an $i \in 1..m$ such that $\mathcal{T}$ contains the axiom $A_{i(j-1)} \doteq \exists P_j$.

Intuitively, if a schema contains an active set H of chains, then in every model of the schema each object that is an instance of all initial concepts of the chains in H requires the existence of a sequence of objects that are connected through the roles of the chains and are instances of successive concepts in the chains. This does not hold if the set of chains is not active, and we can say that the generation of objects along the chains can be blocked at some point. All reductions that we provide are based on the construction of a set of chains whose activation depends on whether the problem we encode has a positive answer or not. In fact, the reductions we give show that the complexity of reasoning on schemata lies precisely in the detection of sets of active chains.

We show **coNP**-hardness of concept consistency in acyclic $\mathcal{AL}$-schemata by a reduction from validity in DNF. Let $f := \{c_1, \ldots, c_n\}$ be a set of clauses, where each clause is a set of literals over the propositional letters $V := \{1, \ldots, m\}$. We call a truth value assignment to the letters of V a V*-assignment*. We construct now an $\mathcal{AL}$-schema $\mathcal{S}_f$ such that a certain concept of $\mathcal{S}_f$ is consistent iff f is valid. $\mathcal{S}_f$ consists of two parts $\mathcal{S}_V$ (which depends only on V) and $\mathcal{S}_c$ (which encodes the clauses).

$\mathcal{S}_V := (\mathcal{A}_V, \mathcal{P}_V, \mathcal{T}_V)$ is defined by $\mathcal{A}_V := \{D_0, \ldots, D_m\} \cup \{E^+_{ij}, E^-_{ij} \mid i \in 1..m, j \in 0..i-1\}$, $\mathcal{P}_V := \{P^+, P^-\}$, and $\mathcal{T}_V$ contains the following axioms:

- For each $i \in 1..m$,

$$\begin{array}{lll} D_i & \doteq & \forall P^+.D_{i-1} \sqcap \exists P^+ \sqcap \forall P^-.D_{i-1} \sqcap \exists P^- \\ D_i & \doteq & \forall P^+.E^+_{i(i-1)} \sqcap \forall P^-.E^-_{i(i-1)}. \end{array} \quad (1)$$

- For each $i \in 2..m, j \in 1..i-1$,

$$\begin{array}{lll} E^+_{ij} & \doteq & \forall P^+.E^+_{i(j-1)} \sqcap \forall P^-.E^+_{i(j-1)} \\ E^-_{ij} & \doteq & \forall P^+.E^-_{i(j-1)} \sqcap \forall P^-.E^-_{i(j-1)}. \end{array}$$

- For each $i \in 1..m, j \in 0..i-1$,

$$E^+_{ij} \doteq \neg E^-_{ij}.$$

The interesting property of this schema is that in every model $\mathcal{I}$ of $\mathcal{S}_V$ with $D^{\mathcal{I}}_m \neq \emptyset$, for every possible V-assignment α there is an object $o_\alpha \in D^{\mathcal{I}}_0$ such that for all $i \in 1..m$ the following holds: $o_\alpha \in E^{+\mathcal{I}}_{i0}$, iff $\alpha(i) =$ **true**, and $o_\alpha \in E^{-\mathcal{I}}_{i0}$, iff $\alpha(i) =$ **false**. This property can be exploited for the reduction of validity as follows. We

encode the clauses of f in such a way that if there is a V-assignment α that does not satisfy f (i.e. in every clause there is at least one literal not satisfied by α) then a certain set H_α of chains corresponding to α becomes active. Additionally, we enforce a contradiction in the schema if we try to populate the final concepts of these chains with a common instance. The axioms in S_V ensure that if we try to populate D_m we are indeed forced to generate for each α a common instance of all initial concepts in the chains of H_α, and therefore generate a contradiction if α does not satisfy f. On the other hand, if f is valid, then there is no set of active chains that forces a contradiction and D_m is consistent in $\mathcal{S}_f$.

$\mathcal{S}_c := (\mathcal{A}_c, \mathcal{P}_c, \mathcal{T}_c)$, which encodes the clauses in the desired way is defined by $\mathcal{A}_c := \{C^+_{ij}, C^-_{ij} \mid i \in 1..m, j \in 0..n\}$, $\mathcal{P}_c := \{P_c\}$, and $\mathcal{T}_c$ contains for each $i \in 1..m$, $j \in 1..n$ the following axioms:

$$C^+_{i(j-1)} \doteq \forall P_c.C^+_{ij} \qquad C^-_{i(j-1)} \doteq \forall P_c.C^-_{ij}$$
$$C^+_{i(j-1)} \doteq \exists P_c, \text{ if } -i \in c_j \qquad C^-_{i(j-1)} \doteq \exists P_c, \text{ if } i \in c_j$$

We can now specify the set H_α of chains corresponding to a V-assignment α, as the set of m P_c-chains that start at C^{*i}_{i0}, where "$*_i$" stands for "$+$" if $\alpha(i) =$ **true**, and for "$-$" if $\alpha(i) =$ **false**. It is not difficult to see that H_α is active iff α does not satisfy f.

By connecting the two schemata $\mathcal{S}_V$ and S_c and adding an axiom that forces a contradiction if a certain concept is forced to be populated, we obtain the schema $\mathcal{S}_f$ that realizes the desired reduction. $\mathcal{S}_f := (\mathcal{A}_f, P_f, \mathcal{T}_f)$ is defined by $\mathcal{A}_f := \mathcal{A}_V \cup \mathcal{A}_c \cup \{F\}$, $\mathcal{P}_f := \mathcal{P}_V \cup \mathcal{P}_c$, and with $\mathcal{T}_f$ obtained by adding to $\mathcal{T}_V \cup \mathcal{T}_c$ the following axioms:

$$E^+_{10} \doteq \exists P_c \qquad C^+_{1n} \doteq \exists P_c \qquad F \doteq \neg F,$$
$$E^-_{10} \doteq \exists P_c \qquad C^-_{1n} \doteq \exists P_c$$

and for each $i \in 1..m$,

$$E^+_{i0} \doteq \forall P_c.C^+_{i0} \qquad C^+_{in} \doteq \forall P_c.F$$
$$E^-_{i0} \doteq \forall P_c.C^-_{i0} \qquad C^-_{in} \doteq \forall P_c.F.$$

Theorem 3 *Let f be a propositional formula in DNF, and $\mathcal{S}_f$ the schema obtained from f as specified above. Then D_0 is consistent in $\mathcal{S}_f$ iff f is valid.*

Corollary 4 *Concept consistency in acyclic $\mathcal{ALU}$-schemata is **coNP**-hard.*

The idea exploited for the previous reduction can be extended in a straightforward way to show **PSPACE**-hardness of reasoning in acyclic $\mathcal{ALU}$-schemata. The result is established by reducing validity of quantified boolean formulae (QBF for short) to concept consistency. Validity of QBFs is a well known **PSPACE**-complete problem, and the hardness result holds already for the restricted case where the propositional part is in DNF, which is the one we consider here. We use a reformulation of the problem similar to the one presented in [19], where a QBF $q := Q.f$ is constituted by a quantifier prefix $Q := (Q_m m) \cdots (Q_1 1)$, and a set f of clauses in DNF over $V := \{1, \ldots, m\}$.

The key observation to our proof lies in the fact that by using disjunction we can replace schema $\mathcal{S}_V$ with a schema $\mathcal{S}_Q$ with the following property: Each model of $\mathcal{S}_Q$ in which D_m is populated corresponds to a set of V-assignments that is canonical for Q. Since we give later an alternative proof of the same hardness result we omit further details.

Theorem 5 *Concept consistency in acyclic $\mathcal{ALU}$-schemata is **PSPACE**-hard.*

In order to show **PSPACE**-hardness of concept consistency in general $\mathcal{AL}$-schemata we use again a reduction from validity of QBFs. However, differently from $\mathcal{ALU}$, in order to handle existential quantifiers in the prefix of the formula, we cannot directly exploit the method based on the generation of objects that correspond to truth assignments. Instead, we make use of the possibility to simulate a binary m-bit counter by means of a cyclic $\mathcal{AL}$-schema. The schema contains $2m$ concepts $B^-_i, B^+_i, i \in 1..m$, representing values 0 and 1 of the m bits, m roles $P_1, \ldots, P_m$, and for $i \in 1..m$ the following axioms:

$$\begin{array}{lll} B^-_i & \doteq & \forall P_1.B^-_i \sqcap \cdots \sqcap \forall P_{i-1}.B^-_i \sqcap \forall P_i.B^+_i \sqcap \exists P_i \\ B^+_i & \doteq & \forall P_1.B^+_i \sqcap \cdots \sqcap \forall P_{i-1}.B^+_i \sqcap \\ & & \forall P_{i+1}.B^-_i \sqcap \cdots \sqcap \forall P_m.B^-_i \sqcap \neg B^-_i \end{array}$$

We have chosen to represent the least significant bit by B^-_1 and B^+_1. If in a model of this schema the concepts $B^-_1, \ldots, B^-_m$ contain a common instance, then for each number n between 0 and $2^m - 1$ there is an object o_n which is a common instance of the concepts that give the binary representation of n. We express the fact that the existence of an object corresponding to a certain number is enforced by saying that the count proceeds till that number.

Given a QBF $q := Q.f$, where Q contains m quantifiers, we embed the chains of schema $\mathcal{S}_c$ that encode the clauses of f in the counter. The idea is that these chains should block the count if there is a set **A** of V-assignments canonical for Q and for which all assignments in **A** satisfy f. If such a set does not exist the count proceeds to the end and a contradiction is enforced. Therefore all sets **A** have to be tried in succession. If a letter i is universally quantified in Q, then each set **A** contains pairs of assignments that assign the same values to $i{+}1, \ldots, m$ and a different value to i. Since both assignments that constitute such a pair have to satisfy f, if q is valid the count for both has to be blocked. Therefore, we insert the two chains corresponding to i in parallel in the schema, which ensures the generation of two different objects that separately activate the count. Conversely, if i is existentially quantified, then the sets **A** come in pairs containing the same assignments except for the value assigned to i by the assignments that correspond each other in the two sets. Therefore we have to try the two sets in succession and we insert the two chains that encode i in succession in the schema, one connected to B^-_i, and the other connected to B^+_i. This ensures that they are both traversed as the count proceeds and can separately contribute to block it. For the full details of the schema $\mathcal{S}'_q$ that implements the desired reduction we refer to [7], and we state here only the final result.

Theorem 6 *Let q be a QBF and $\mathcal{S}'_q$ the cyclic $\mathcal{AL}$-schema obtained from q. $\mathcal{S}'_q$ contains a distinguished concept name D_0, and D_0 is consistent in $\mathcal{S}'_q$ iff q is valid.*

Corollary 7 *Concept consistency in general-$\mathcal{AL}$-schemata is **PSPACE**-hard.*

3.3 Eliminating existential quantification

In [4] it is shown that verifying concept consistency in $\mathcal{AL}$-schemata is **coNP**-hard. The result is established by exploiting a reduction from subsumption between concept expressions in $\mathcal{ALE}$ (which was shown **coNP**-complete in [13]) to concept consistency in $\mathcal{AL}$-schemata. The reduction is based essentially on the elimination of existential quantification from an $\mathcal{ALE}$ concept containing a single role by introducing suitable axioms in the schema.

The elimination of existential quantification can easily be generalized to schemata and to an arbitrary number of roles. It proves useful

both as a theoretical tool to show complexity results, and as a method to exploit reasoning techniques developed for logics without existential quantification, for their more expressive variants which feature this constructor.

For a normalized $\mathcal{ALC}$-schema $\mathcal{S}_\mathcal{E} := (\mathcal{A}_\mathcal{E}, \mathcal{P}_\mathcal{E}, \mathcal{T}_\mathcal{E})$ we define the schema $\mathcal{S} := (\mathcal{A}, \mathcal{P}, \mathcal{T})$ obtained from $\mathcal{S}_\mathcal{E}$ by *$\mathcal{E}$-elimination*:

- $\mathcal{A} := \mathcal{A}_\mathcal{E}$.
- For a role $P \in \mathcal{P}_\mathcal{E}$, let $\mathcal{P}_P := \{P_A \mid \exists P.A \text{ appears in } \mathcal{T}_\mathcal{E}\}$. Then $\mathcal{P} := \mathcal{P}_\mathcal{E} \cup \bigcup_{P \in \mathcal{P}_\mathcal{E}} \mathcal{P}_P$.
- $\mathcal{T}$ is obtained from $\mathcal{T}_\mathcal{E}$ by substituting each axiom

  $A \doteq \exists P.B$ with $A \doteq \exists P_B \sqcap \forall P_B.B$, and
  $A \doteq \forall P.B$ with $A \doteq \forall P.B \sqcap \forall P_{A_1}.B \sqcap \cdots \sqcap \forall P_{A_n}.B$, where $\mathcal{P}_P = \{P_{A_1}, \ldots, P_{A_n}\}$.

We observe that the number of new roles introduced in $\mathcal{S}$ is linear in the number of axioms in $\mathcal{S}_\mathcal{E}$, and that the size of $\mathcal{S}$ is at most quadratic in the size of $\mathcal{S}_\mathcal{E}$. Although we have replaced each qualified existential quantification with an unqualified one on a different role, the interaction between universal and existential quantification is still captured correctly. In fact, the following theorem holds.

Theorem 8 *Let $\mathcal{S}_\mathcal{E}$ be an $\mathcal{ALC}$-schema and $\mathcal{S}$ the schema obtained from $\mathcal{S}_\mathcal{E}$ by $\mathcal{E}$-elimination. A concept A is consistent in $\mathcal{S}$ iff it is consistent in $\mathcal{S}_\mathcal{E}$.*

We obtain immediately that reasoning on $\mathcal{ALE}$-schemata can be reduced to reasoning on $\mathcal{AL}$-schemata, and from Theorem 2 a new upper bound for reasoning on acyclic $\mathcal{ALC}$-schemata.

Corollary 9 *Concept consistency in $\mathcal{ALE}$-schemata is **coNP**-complete for acyclic schemata and **PSPACE**-complete for general schemata.*

Corollary 10 *Concept consistency in acyclic $\mathcal{ALC}$-schemata can be decided in worst case polynomial space in the size of the schema.*

We get also a further confirmation of the fact that the complexity source introduced by (acyclic) axioms is of the same nature as the one introduced by existential quantification, and results precisely from the interplay between (unqualified) existential quantification and universal quantification.

$\mathcal{E}$-elimination allows us also to show lower bounds for reasoning on $\mathcal{ALU}$-schemata, by exploiting the lower bounds for $\mathcal{ALC}$. In particular, from **PSPACE**-hardness of concept consistency in acyclic $\mathcal{ALC}$-schemata (which follows immediately from the hardness result for $\mathcal{ALC}$ proved in [19]), by applying Theorem 8 we obtain an alternative proof of Theorem 5. From **EXPTIME**-completeness of reasoning on general $\mathcal{ALC}$-schemata [5], we obtain the same completeness result for reasoning on general $\mathcal{ALU}$-schemata, of which the hardness part was open.

Corollary 11 *Concept consistency in general $\mathcal{ALU}$-schemata is **EXPTIME**-hard.*

4 CONCLUSIONS

Both in databases and in knowledge representation, reasoning on schemata is of fundamental importance and this paper represents the first systematic investigation of the issues related to its computational complexity. We have analyzed the complexity of reasoning on primitive schemata for the relevant class of description logics below $\mathcal{ALC}$. Three separate technical contributions are given: (1) A method to reason on acyclic $\mathcal{ALU}$-schemata in polynomial space. (2) Direct reductions that provide tight lower bounds for reasoning on $\mathcal{AL}$-schemata and acyclic $\mathcal{ALU}$-schemata. (3) A general technique to eliminate the constructor for existential quantification. Applied together they allow us to characterize completely the computational complexity of reasoning in all considered languages, both with respect to acyclic and general schemata.

The present work can be extended in various directions. First, one can study the impact on computational complexity of augmenting the expressivity of the logic used to express the schema. Particularly important in this context is the addition of inverse roles, and first steps in this direction are done in [4, 10]. Second, one can study heuristics to reduce the computational complexity of reasoning under assumptions that are reasonable in practice and that are common in databases, following e.g. the ideas given in [9].

REFERENCES

[1] F. Baader, 'Terminological cycles in KL-ONE-based knowledge representation languages', in *Proc. of AAAI-90*, pp. 621–626, (1990).

[2] F. Baader, 'Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles', in *Proc. of IJCAI-91*, (1991).

[3] A. Borgida, 'Description logics in data management', *IEEE Trans. on Knowledge and Data Engineering*, **7**(5), 671–682, (1995).

[4] M. Buchheit, F. Donini, W. Nutt, and A. Schaerf, 'A refined architecture for terminological systems: Terminology = schema + views', Technical Report RR-95-09, DFKI, (1995).

[5] M. Buchheit, F. Donini, and A, Schaerf, 'Decidable reasoning in terminological knowledge representation systems', *J. of Artificial Intelligence Research*, **1**, 109–138, (1993).

[6] M. Buchheit, M. Jeusfeld, W. Nutt, and M. Staudt, 'Subsumption between queries to Object-Oriented databases', *Information Systems*, **19**(1), 33–54, (1994).

[7] D. Calvanese, *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms*, Ph.D. dissertation, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 1996.

[8] D. Calvanese, G. De Giacomo, and M. Lenzerini, 'Structured objects: Modeling and reasoning', in *Proc. of DOOD-95*, number 1013 in LNCS, pp. 229–246, (1995).

[9] D. Calvanese and M. Lenzerini, 'Making object-oriented schemas more expressive', in *Proc. of PODS-94*, pp. 243–254, (1994).

[10] D. Calvanese, M. Lenzerini, and D. Nardi, 'A unified framework for class based representation formalisms', in *Proc. of KR-94*, (1994).

[11] G. De Giacomo, *Decidability of Class-Based Knowledge Representation Formalisms*, Ph.D. dissertation, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 1995.

[12] G. De Giacomo and M. Lenzerini, 'Concept language with number restrictions and fixpoints, and its relationship with μ-calculus', in *Proc. of ECAI-94*, pp. 411–415, (1994).

[13] F. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt, 'The complexity of existential quantification in concept languages', *AIJ*, **2–3**, 309–327, (1992).

[14] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt, 'Tractable concept languages', in *Proc. of IJCAI-91*, pp. 458–463, (1991).

[15] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt, 'The complexity of concept languages', Technical Report RR-95-07, DFKI, (1995).

[16] B. Nebel, 'Terminological reasoning is inherently intractable', *AIJ*, **43**, 235–249, (1990).

[17] B. Nebel, 'Terminological cycles: Semantics and computational properties', in *Principles of Semantic Networks*, ed., John F. Sowa, 331–361, Morgan Kaufmann, Los Altos, (1991).

[18] K. Schild, 'Terminological cycles and the propositional μ-calculus', in *Proc. of KR-94*, pp. 509–520, (1994).

[19] M. Schmidt-Schauß and G. Smolka, 'Attributive concept descriptions with complements', *AIJ*, **48**(1), 1–26, (1991).